PowerSensor 2: a Fast Power Measurement Tool

John W. Romein and Bram Veenboer
ASTRON (Netherlands Institute for Radio Astronomy), PO Box 2, 7990 AA Dwingeloo, The Netherlands
Email: {romein,veenboer}@astron.nl

***Abstract*—*PowerSensor 2* is a tool that measures the instantaneous power consumption of PCIe cards and SoC development boards like GPUs, Xeon Phis, FPGAs, DSPs, and network cards, at sub-millisecond time scale. It consists of a commodity microcontroller, commodity current sensors, and (for PCIe devices) a PCIe riser card. The microcontroller reports measurements to the host via USB. A small host library assists an application to determine its own energy efficiency. The high time resolution (up to 8.62 kHz) provides much better insight into energy usage than low-resolution built-in power meters (if available at all), as PowerSensor 2 enables analysis of individual compute kernels.**

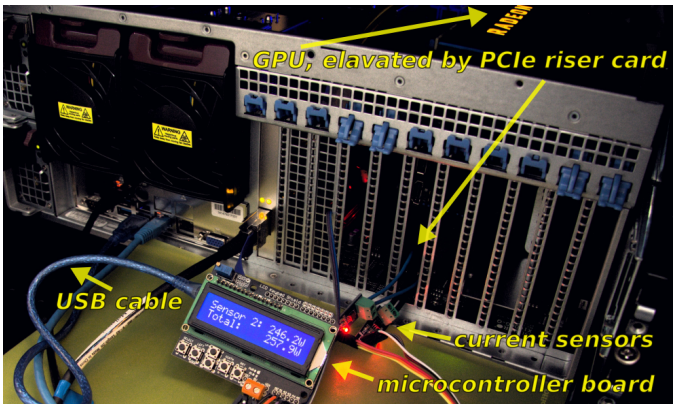

Fig. 1. A PowerSensor measures the instantaneous power use of a GPU.

Introduction

PowerSensor 2 is a low-cost, custom-built device that measures the instantaneous power consumption of GPUs and other (peripheral) devices at a high time resolution. It consists of an Arduino Leonardo (or Arduino Pro Micro) microcontroller board, current sensors (ACS712), a PCIe riser card (to measure the power drawn from the motherboard), an optional LCD screen, and a USB cable that is connected to the host. Fig. 1 shows a typical use case of a PowerSensor attached to a GPU. In this scenario, we use three sensors that measure the power drawn through the PCIe slot (12 V and 3.3 V) and the external PCIe cable. As the microcontroller has only one ADC, it reads the sensors one after another, and reports the measurements via USB to the host.

The tool has been used successfully to analyze several applications on PCIe devices like GPUs, a Xeon Phi, a 40 GbE network card, and an FPGA, as well as SoC development platforms like the Jetson TX1 and an EVMK2H DSP board [1–3]. The firmware, host library, support programs, and how-to-build-it-yourself manual are available for download [4].

```
class PowerSensor {
  public:
    ...
    State read();
};

double Joules(const State &first, const State &second);
double Watt(const State &first, const State &second);
double seconds(const State &first, const State &second);
```

Listing 1. PowerSensor host library interface.

Operation modes

PowerSensor supports two operation modes.

With ***interval-based measurements***, an application measures the power use of a device during some time interval. At the start and at the end, the application invokes a library function that returns an object that represents the instantaneous power state. With these two objects, the application asks the library how much energy or time was spent during the interval (in Joules, Watt, or seconds). An application can then determine its own energy efficiency by dividing the number of operations (obtained by profiling or analytically) by the measured energy use. The library interface (listing 1) is easy to use (listing 2).

As *GPU* kernels are typically launched asynchronously by enqueuing them to some stream, the PowerSensor state must be read by a callback function that is invoked whenever the GPU kernel starts or stops executing. Both CUDA and OpenCL support these callback functions.

In ***continuous measurements*** mode, PowerSensor writes a stream of consecutive measurements to file. The library starts a low-overhead thread that runs asynchronously with the application, and writes tuples of current time and wattage to file (in ASCII), 8,620 times per second. The library allows the application to put markers in this file, e.g., to annotate an event such as the start of a particular kernel execution. These markers can be cross-correlated with the power measurements. The file can be easily used to create time-vs.-power graphs by any plotting tool that allows ASCII input, as shown below.

Modifying the application source code to use the library is not obligatory; the included `psrun` utility can monitor the power use of a device during the execution of an unmodified

```
int main()
{
  PowerSensor sensor("/dev/ttyACM0");
  State start = sensor.read();
  ... // do work, e.g. on GPU
  State stop = sensor.read();
  cout << "It used " << Joules(start, stop) << 'J' << endl;
}
```

Listing 2. Example use of the host library.

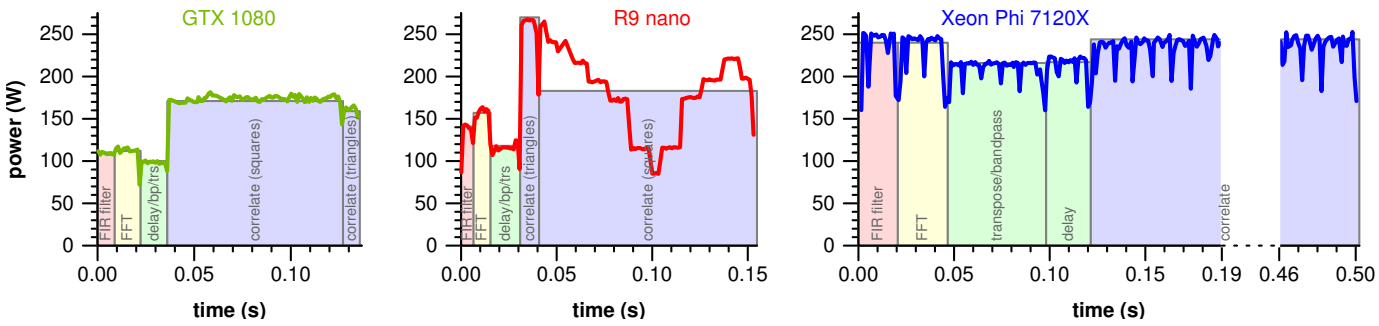

Fig. 2. Continuous power measurements of several devices.

application. This utility also supports the continuous measurements mode.

Installation and Use of This Tool

The installation of this tool requires some basic skills in electronics, but is not excessively difficult. Once installed, its use is simple. The number and types of the used current sensors, the voltages of the power lines, and calibration weights are easily configurable using the `psconfig` utility.

The Hall-effect current sensors must be calibrated for the local magnetic field. To measure very low currents, an ACS712 sensor board with integrated voltage multiplier can be used. A PCIe riser card of sufficient quality is needed to maintain the PCIe signal integrity; we use the Adexelec PEXP16-EX.

As we do not measure voltages, the power supply voltage must be stable, also under varying load. The sensor and ADC error tolerances add up to 3.7%, but with proper calibration, our measurements are typically within 1% of built-in GPU power meters and lab equipment.

PowerSensor 2 achieves 9 times better time resolution than PowerSensor 1, by using another microcontroller with an integrated USB port, and by improving the firmware. The time resolution is 116 µs times the number of attached sensors, and is limited by the ADC conversion time.

Examples of Insights Obtained with PowerSensor

PowerSensor gives insight into an application's power efficiency, as illustrated by Fig. 2. This radio-astronomical pipeline filters, corrects, and correlates the signals from 960 receivers [2]. The figure shows the instantaneous power consumption of three different devices. The correlation between energy use and executed kernels is clearly visible. The shaded area below the curve corresponds to the total energy used by a kernel. On the NVIDIA GTX 1080 GPU (left), some kernels draw much more power than others.

The AMD R9 nano GPU (middle) has a Thermal Design Power (TDP) of 175 W. However, the graph shows temporary power consumption as high as 275 W. After 13 ms, the device starts throttling, stepwise reducing power usage to as low as 85 W to compensate for the excess power usage, then jumping back to 220 W. The long-term average power consumption is indeed 175 W. The performance of one kernel depends strongly on the power usage of the other kernels: the correlate-triangles function runs at a high clock frequency and thus a high power consumption because the first three functions did not use their full power budget. A high-time-resolution tool like PowerSensor is indispensable to analyze this behavior.

The graph for the Xeon Phi 7120X (right) shows repetitive power dips at a 100 Hz rate. The Linux kernel periodically interrupts all cores, during which much less energy is drawn than when the application performs heavy vector computations. We discovered this behavior with PowerSensor; we did not notice it when profiling the application with VTune Amplifier.

Related Work

There are many power-measurement tools that bear some resemblance. They are all elegant in some aspects, but none of them combines all advantages of high time resolution, simplicity, low cost, availability, and full library support. PowerInsight [5] measures both voltages and currents, but has lower time resolution and does not support the interval-based mode described above. PowerMon 2 [6] uses a well-designed but difficult to obtain custom PCB; it also cannot handle 150 W PCIe power cables. Ilsche et al. present a highly accurate but costly and complex method [7]. Others have built their own power measurement tools, for example, to validate a power estimation framework for GPUs (GPUSimPow) [8], or to analyze the power behavior of the Xeon Phi [9], but these tools are not publicly available.

Complementary to PowerSensor are tools that measure the energy consumption of the host CPU and DRAM, like LIKWID [10], PAPI [11], Linux perf, and Intel PCM.

Conclusion

The high time resolution, low cost, ease of use, and public availability make PowerSensor 2 a useful tool for power measurements of (peripheral) devices.

Acknowledgements

This work received funding from the Netherlands Organization for Scientific Research (NWO) through the DAS-5 [12] and Triple-A (617.001.204) grants, the Dutch Ministry of EZ and the province of Drenthe through the ASTRON-IBM Dome grant, and the European Commission through the H2020-FETHPC DEEP-EST grant (Grant Agreement nr. 754304).

REFERENCES

[1] P. Lenkiewicz, P. C. Broekema, and B. Metzler, "Energy-efficient data transfers in radio astronomy with software UDP RDMA," *Future Generation Computer Systems*, March 2017, available online.

[2] J. W. Romein, "A Comparison of Accelerator Architectures for Radio-Astronomical Signal-Processing Algorithms," in *Int. Conf. on Parallel Processing (ICPP'16)*, Philadelphia, PA, August 2016, pp. 484–489.

[3] B. Veenboer, M. Petschow, and J. W. Romein, "Image-Domain Gridding on GPUs," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS'17)*, Orlando, FL, May 2017, pp. 545–554.

[4] https://gitlab.com/astron-misc/PowerSensor.

[5] J. H. Laros III, P. Pokorny, and D. DeBonis, "PowerInsight – A Commodity Power Measurement Capability," in *Int. Workshop on Power Measurement and Profiling*, Arlington, VA, June 2013.

[6] D. Bedard, M. Y. Lim, R. Fowler, and A. Poterfield, "PowerMon: Fine-grained and Integrated Power Monitoring for Commodity Computer Systems," in *Proc. of the IEEE SoutheastCon*, Charlotte-Concord, NC, March 2010, pp. 479–484.

[7] T. Ilsche, D. Hackenberg, S. Graul, J. Schuchart, and R. Schöne, "Power Measurements for Compute Nodes: Improving Sampling Rates, Granularity and Accuracy," in *International Green Computing Conference and Sustainable Computing Conference (IGSC'15)*, Dec. 2015, pp. 1–8.

[8] J. Lucas, S. Lal, M. Andersch, M. Alvarez-Mesa, and B. Juurlink, "How a Single Chip Causes Massive Power Bills GPUSimPow: A GPGPU Power Simulator," in *IEEE Int. Symposium on Performance Analysis of Systems and Software (ISPASS'13)*, Austin, TX, April 2013, pp. 97–106.

[9] F. D. Igual *et al.*, "A Power Measurement Environment for PCIe Accelerators: Application to the Intel Xeon Phi," *Computer Science – Research and Development*, vol. 30, no. 2, pp. 115–124, May 2015.

[10] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A Lightweight Performance-oriented Tool Suite for x86 Multicore Environments." in *Int. Conf. on Parallel Processing Workshops (ICPPW'10)*, San Diego, CA, September 2010, pp. 207–216.

[11] V. M. Weaver et al., "PAPI 5: Measuring Power, Energy, and the Cloud," in *IEEE Int. Symposium on Performance Analysis of Systems and Software (ISPASS'13)*, Austin, TX, April 2013, pp. 124–125.

[12] H. Bal *et al.*, "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term," *IEEE Computer*, vol. 49, no. 5, pp. 54–63, May 2016.